

Package ‘g.data’

October 13, 2022

Version 2.4

Date 2013-12-16

Title Delayed-Data Packages

Author David Brahm <brahm@alum.mit.edu>

Maintainer David Brahm <brahm@alum.mit.edu>

Description Create and maintain delayed-data packages (ddp's). Data stored in a ddp are available on demand, but do not take up memory until requested. You attach a ddp with `g.data.attach()`, then read from it and assign to it in a manner similar to S-PLUS, except that you must run `g.data.save()` to actually commit to disk.

License GPL

NeedsCompilation no

Repository CRAN

Date/Publication 2013-12-17 07:09:33

R topics documented:

`g.data.save` 1

Index 4

`g.data.save` *Create and Maintain Delayed-Data Packages*

Description

`g.data.save` reads the data in search position "pos", and writes them as a delayed-data package ("DDP") to "dir". Data objects are initially created as promise objects, the promise being to load the data and return it the first time the item is requested.

`g.data.attach` attaches such a package, in position 2 by default.

Usage

```

g.data.attach(dir, pos=2, warn=TRUE, readonly=FALSE)
g.data.save(dir=attr(env, "path"), obj=ls(env, all.names=TRUE),
            pos=2, rm.obj=NULL)
g.data.get(item, dir)
g.data.put(item, value, dir)

```

Arguments

<code>dir</code>	Directory (full pathname) of DDP.
<code>pos</code>	Search path position.
<code>warn</code>	Logical: warn user if directory being attached doesn't exist
<code>readonly</code>	Logical: set an attribute on the package that will cause <code>g.data.save</code> to abort.
<code>obj</code>	Object name(s).
<code>rm.obj</code>	Objects to remove, both in memory and on disk.
<code>item</code>	Item to retrieve from an unattached package.
<code>value</code>	Value for the data item being put with <code>g.data.put</code> .

Details

Data stored in a delayed-data package (DDP) are available on demand, but do not take up memory until requested. You attach a DDP with `g.data.attach`, then read from it and assign to it via its position on the search path (similar to `S-Plus`). Unlike `S-Plus`, you must run `g.data.save()` to actually commit to disk.

You can create a DDP from any position in the search path, not just one created with `g.data.attach`; e.g. you can attach a list or dataframe, and its components will become objects in the DDP. In this case, the call to `g.data.save(dir)` must specify the path where files will be saved. If the DDP was created with `g.data.attach`, then its directory is known and does not need to be passed again to `g.data.save`.

The filename associated with an object 'obj' is 'obj.RData', except that uppercase letters are preceded by an '@' symbol. This is required by Windows since 'x.RData' and 'X.RData' are the same file under that OS. Unexported functions `g.data.mash` and `g.data.unmash` perform the object name / filename conversion, e.g. `g.data.mash(dir, "aBcD")` returns "dir/a@Bc@D.RData".

`g.data.get` can be used to get a single piece of data from a package, without attaching the package. `g.data.put` puts a single item into an unattached package.

Value

`g.data.get` returns the requested data.

See Also

[delayedAssign](#)

Examples

```
## Not run:
ddp <- tempfile("newdir")          # Where to put the files
g.data.attach(ddp)                 # Warns that this is a new directory
assign("m1", matrix(1, 5000, 1000), 2)
assign("m2", matrix(2, 5000, 1000), 2)
g.data.save()                      # Writes the files
detach(2)

g.data.attach(ddp)                 # No warning, because directory exists
ls(2)
system.time(print(dim(m1)))        # Takes time to load up
system.time(print(dim(m1)))        # Second time is faster!
find("m1")                         # m1 still lives in pos=2, is now real
assign("m3", m1*10, 2)
g.data.save()                      # Or just g.data.save(obj="m3")
detach(2)

mym2 <- g.data.get("m2", ddp)      # Get one objects without attaching
unlink(ddp, recursive=TRUE)        # Clean up this example

## End(Not run)

## Not run:
ddp <- tempfile("newdir")          # New example
y <- list(m1=1:1000, m2=2:1001)
attach(y)                          # Attach an existing list or dataframe
g.data.save(ddp)
detach(2)
unlink(ddp, recursive=TRUE)        # Clean up this example

## End(Not run)
```

Index

* **data**

g.data.save, [1](#)

delayedAssign, [2](#)

g.data.attach(g.data.save), [1](#)

g.data.get(g.data.save), [1](#)

g.data.put(g.data.save), [1](#)

g.data.save, [1](#)